Bjorn Johnson

CSCI 267

Fall 2014

Final Paper

For this project I intended to create a wifi enabled remote to control a home air conditioner (or other IR controlled devices) via an iphone app. I was able to achieve this with a few changes to my original plans.

My hopes at the beginning were to decode the infrared on my own, using no other libraries. I was able to decode IR signals by measuring the incoming 38khz on pulses vs the delay in between. Unfortunately something in my timing was off (we are dealing with signals in the microsecond range) and I was not able to reliably retransmit the signal. My best guess is that the timing would be off due to the angle of the original transmission (from a remote) and that it would then throw the retransmission off.

Also, when collecting raw IR signal it is difficult to tell when the transmission ends and restarts (if you hold down the power button it will keep transmitting signal that may not be useful). A final difficulty was in the fact that there at least 4 well documented ways of transmitting IR (SONY, NEC, RC5, and RC6) as well as many undocumented ways. Even within a specific protocol such as NEC, the are variations on the way the code is repeated when a button is held down.

To solve this problem I used Ken Shirrif's IR library[1] to do much of the heavy lifting. With this library I was easily able to decode and retransmit NEC as well as Sony codes. The library has specifications which it can match signal to. As long as a code is within a certain tolerance, it is able to match it to a specific protocol for retransmission.

Although very helpful, this IR library does have some limitations. First, it was not easy to switch between protocols on the fly. As it stands right now, I can only decode one protocol at a time. In a perfect world it would be nice to decode any remote I have without having to change my code. Second, the library itself also took up a fair amount of space, limiting the extra code I could write in the sketch. With the IR Library and the CC3000 libraries, I was very close to capacity for the Uno.

The other major difference from my original proposal is that I chose to use Adafruit's web server[2] library rather than the iArduino iphone app. In the end I did not feel that the iArduino fit my needs as well as I had originally thought, as it mostly used sliders (range of values) and toggles (on or off states) to control the Arduino, rather than buttons (send a signal once). The Adafruit server was also free, so it fit nicely into my budget.

The server, as well as the shield itself, was also a bit more difficult to work with than I thought. The shield itself has severe limitations in the number of connections it can handle at once or even in succession. If you happen to send a signal to the server too soon after the last one, it will cause the server to freeze. Also, when starting up, the wifi shield will fail to initialize on a regular basis, which means that you cannot connect to a router. I researched the problem, and it seems to be a limitation either with the TI chip that controls the shield, or the TI library that controls the chip. Unfortunately, these were both beyond my knowledge to fix at the current time. Hopefully revision 2 will be more stable.

In conclusion, after some hiccups, I am very happy to say that I was able to complete the project. I now have a way to control my a/c remotely, and can “pre-cool” my room while I’m away.

Usage:

The device has two modes, “receive” and “transmit”, which a user can switch between using a toggle switch. When the device is powered up, it will automatically connect to a predefined wireless network. Once fully connected it will signal that it is ready to decode IR by light the first red LED on the top of the device. The user will then press power on the remote they wish to capture. When the signal is done being captured, a green light indicates that the process is finished. The user can then proceed to capture up and down signals in a similar fashion. Once the receive phase is complete, the user can then switch to transmit mode. Once in transmit mode, the user connects to the arduinos web address through a web browser, followed by the appropriate code. Ex. 192.168.1.2/power will transmit the power signal. The web page will display the current room temperature as well as the signal just transmitted.

PARTS LIST:

- Arduino Uno

- Adafruit CC3000 WiFi Shield (internet connection)
- 5 x 5mm 1.7V 20mA 800mcd Wide Angle Red Leds
  - 3 x capture start indicators
  - 1 x transmit indicator
  - 1 x receive indicator
- 5 x 180 ohm resistors(for red leds)
- 3 x 5mm 2.1V 30mA 630mcd Green Leds
  - for capture complete indicators
- 3 x 100 ohm resistors (for green leds)
- 9-12V ac adapter(for powering the device)
- 1 x IR Receiver Diode - TSOP38238(for capturing signal)
- 1 x 1.5VDC 50mA 840-850nm IR Led(for transmitting signal)
- 1 x Waterproof DS18B20 Digital temperature sensor (for getting room temp)
- 4.7k ohm resistor for temp sensor
- wiring breadboard
- a whole mess of wires
- toggle switch(to switch between transmit and receive)
- 1 bike light(for when you accidentally burn out one of your red leds)
- WD MyBook case to house everything

LIBRARY LIST

- Ken Shirrifs IR Library
- Adafruit_CC3000.h(for wifi Shield)
- SPI.h(for wifi Shield)
- utility/debug.h(for wifi Shield)
- utility/socket.h(for wifi Shield)
- IRremote.h(from Ken Shirrif)
- OneWire.h(for temp sensor)

Sources

[1] Ken Shirrifs IR Library

http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html

[2] adafruit web server

https://github.com/adafruit/Adafruit_CC3000_Library

[3] One wire sample code (for temperature sensor)

https://www.pjrc.com/teensy/td_libs_OneWire.html

http://playground.arduino.cc/Learning/OneWire